Intel® C++ Compiler 7.0 for Windows* Compatibility with Microsoft* Visual C++* 6.0 and Visual C++ .NET*

Table of contents

Overview

The Intel® C++ Compiler 7.0 for Windows* supports the extensions of Microsoft* Visual C++* to the C and C++ languages. It is integrated into the Visual C++ 6.0 IDE and Visual C++ .NET* IDE. The Intel C++ Compiler specializes in generating optimized code for Intel® IA-32 and Itanium® architecture-based systems.

This document highlights the compatibility of the Intel C++ compiler with Microsoft Visual C++ 6.0 and Microsoft Visual C++ .NET. For more details, please read the section titled *Microsoft Compatibility* in the *Intel C++ Compiler User's Guide*.

Visual C++ 6.0 Compatibility

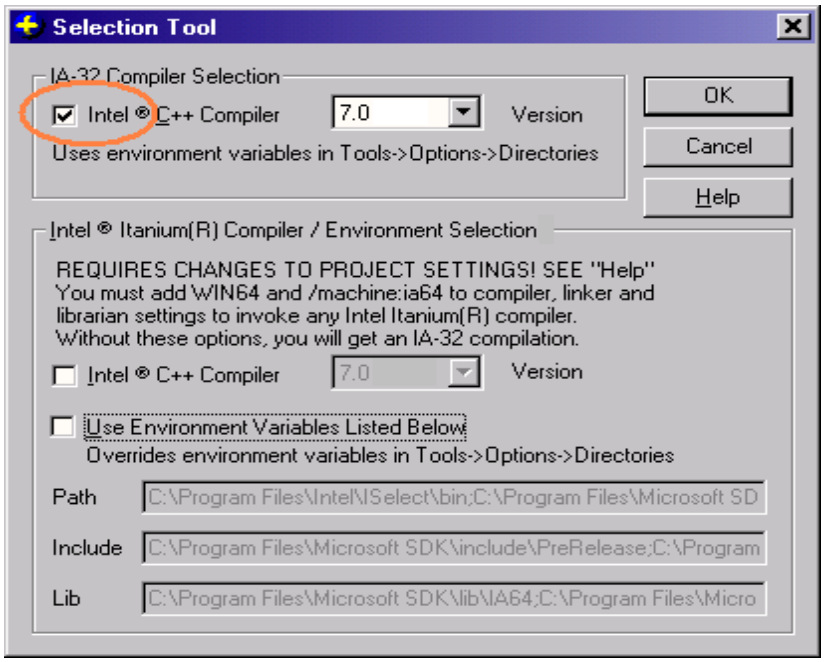
The Intel C++ compiler is **source and binary compatible** with the Microsoft Visual C++ compiler. This allows you to take advantage of the performance enhancement features of the Intel C++ compiler by rebuilding only part of an application with the Intel® compiler. You can mix and match object files and libraries built with the two compilers if required.

The Intel C++ Compiler is integrated into Visual C ++ 6.0 IDE through the `Selection Tool` that is available from the `[Tools\Select Compiler]` menu. Binaries built with the Intel C++ Compiler can be debugged from within the Visual C ++ 6.0 IDE.

Invoking the Intel C++ Compiler through Visual C++ 6.0

To invoke the Intel C++ Compiler from Visual C ++ 6.0 IDE and build IA-32 applications, follow the steps below.

- Open the `Selection Tool` dialog box by choosing `[Select Compiler]` from the `[Tools]` menu.
- To choose the Intel C++ Compiler, click the check-box. If you have more than one version of the Intel C++ Compilers installed, please select the version of your choice.
- Click [OK].
- Rebuild your application with Visual C++ IDE.



To switch back to the Microsoft Compiler, follow the same steps as above and uncheck the Intel Compiler box.

If you want to build only part of your application with the Intel C++ Compiler, follow the steps below:

- Identify the files that need to be compiled with the Intel C++ Compiler
- For each of the files, open the corresponding `Project Settings` dialog box and add **_USE_INTEL_COMPILER** to the pre-processor definitions
- Open the `Selection Tool` window to make sure that the Intel C++ Compiler is **NOT** selected
- Rebuild the application

If you want to build only part of your application with Microsoft Visual C++ Compiler, following the steps below:

- Identify the files that need to be compiled with Microsoft Visual C++ Compiler
- For each of the files, open the corresponding `Project Settings` dialog box and add **_USE_NON_INTEL_COMPILER** to the pre-processor definitions
- Open the `Selection Tool` dialog box to make sure that the Intel C++ Compiler **IS** selected
- Rebuild the application

Programs can be targeted for either IA-32 based systems or Itanium-based systems. Please refer to the document titled *Getting Started with the Intel® C++ Compiler 7.0 for Windows* for more details. Please refer to *Optimizing Applications with the Intel® C++ and Fortran Compilers for Windows* and Linux** for the optimization options supported by the Intel compiler.

Limitations in compatibility of the Intel C++ Compiler with Visual C++ 6.0

Unsupported Compiler Options

The Intel C++ Compiler supports most of the options in Microsoft Visual C++ 6.0 compiler. However the following options are not supported. Most of the unsupported options are useful during development time, and are typically not required to build a working application.

Option	Description
`-Fd`	Name of the PDB file used for debug information for specified source files
`-Gi`	Enable incremental compilation
`-ZI`	Edit and continue debugging (similar effect as the `-Zi` option)
`-Gm`	Enable minimal rebuild
`-QIF`	Emit FPO (frame pointer optimization) records
`-Yd`	Put debug information in every object (Microsoft PCH-specific option)
`-Zm`*n*	Control of maximum memory allocated by the compiler

Unsupported pragmas

The following pragmas are accepted without error but have no effect:

• auto_inline	• inline_depth
• component	• inline_recursion
• function	• intrinsic
• include_alias	• setlocale

Differences in PCH Support

The PCH information generated by the Intel C++ Compiler is not compatible with the PCH information generated by the Microsoft Visual C++ compiler. Please note the following:

- The Intel compiler does not recognize a precompiled header file generated by the Microsoft compiler. If no suitable or recognized precompiled header file exists, compilation proceeds without use of precompiled header files.
- The Microsoft compiler aborts with an error message when it tries to use a precompiled header file generated by the Intel compiler.
- Precompiled header files coexistence: the `-Qpchi` option causes the Intel C++ Compiler to name its PCH files with a .pchi filename suffix and reduces build time. The `-Qpchi` option is on by default, use `-Qpchi-` to turn it off.

For more on compatibility information, please read the section *Microsoft Compatibility* in the *Intel C++ Compiler User's Guide*.

Visual C++ .NET Compatibility

The Intel C++ compiler 7.0 for Windows is integrated into Microsoft Visual C++ .NET IDE. The following capabilities are enabled by the integration:

- Use the Intel C++ Compiler to build new or existing Visual C++ projects
- Select either the Intel C++ Compiler or the Microsoft Visual C++ compiler to build each configuration of a project, or to build individual files of a project.
- Use compilation options specific to the Intel C++ Compiler to optimize your application.
- Select which version of the Intel C++ Compiler to use if multiple versions are installed on your system.

But the support for Visual C++ .NET features is limited to those features required to build Visual C++ 6.0 projects. The details are:

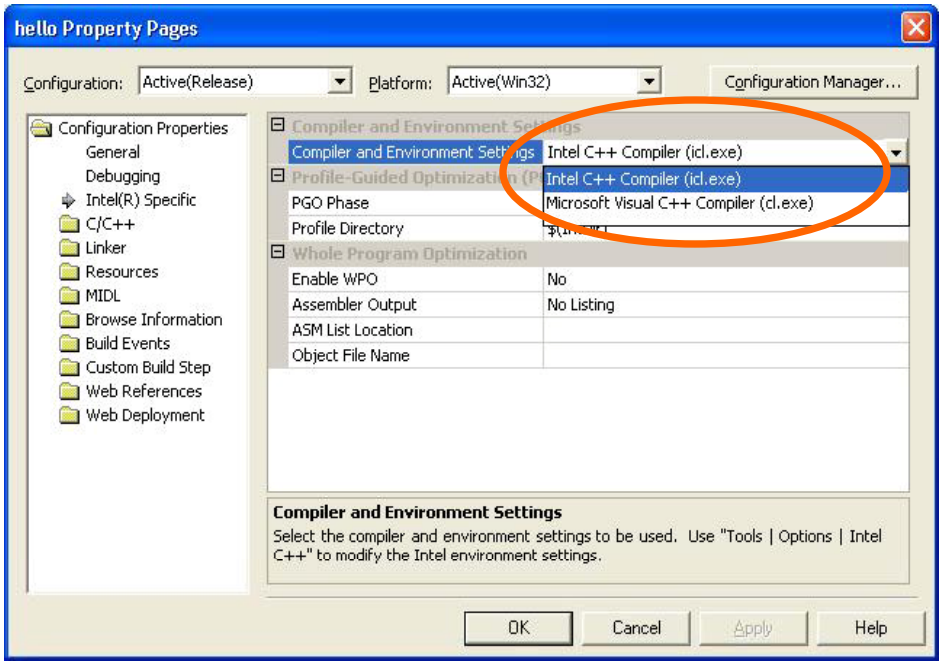
- The Intel C++ Compiler supports building of Visual C++ 6.0 projects that have been converted to Visual C++ .NET projects, provided no new features of Visual C++ .NET are used. If your project uses new features of Visual C++ .NET, it may not build when the Intel C++ Compiler is selected. The unsupported features are listed below in the *Limitations in Compatibility* section.
- Certain compiler options used by the default Visual C++ .NET project configurations may cause the Intel C++ Compiler to display remarks or warnings about unknown or unsupported options. These are also listed in the *Limitations in Compatibility* section below.

Invoking the Intel C++ Compiler through Visual C++ .NET

The Intel C++ compiler is source and binary (native code, not managed code) compatible with the Visual C++ .NET compiler. So you can build either the full project with Intel C++ Compiler or part of the project files with Intel C++ Compiler.

To invoke the Intel C++ Compiler from Visual C ++ .NET IDE, follow the steps below to build IA-32 applications.

- Open the project property window.
- Select "`Intel® Specific`", then select "`Intel C++ Compiler (icl.exe)`" from "`Compiler and Environment Settings`" drop-down list.
- Click `[OK]`.
- Rebuild your application with Visual C++ IDE.



Special macros "**_USE_INTEL_COMPILER**" and "**_USE_NON_INTEL_COMPILER**" work the same way for the Intel C++ Compiler within Visual C++ .NET. However, with the Intel C++ Compiler 7.0, you may select the compiler for each source file using the project property window. The steps are:

a) Select one or more files in the Solution Explorer.
b) Right-click and select Properties to open the project properties window.
c) Select "`Intel® Specific`" and then select the compiler you wish to use to compile the selected file(s) from "`Compiler and Environment Settings`" drop-down list.

The only supported target from Intel C++ Compiler 7.0 with Visual C++ .NET environment is IA-32 systems. There is no support in this release for targeting Itanium-based systems from Visual C++ .NET environment. Please refer to the document titled *Getting Started with the Intel® C++ Compiler 7.0 for Windows* for more details.

Limitations in Compatibility of the Intel C++ Compiler with Visual C++ .NET

The following new Visual C++ .NET compiler options, features, and preprocessor features are not supported by this release of the Intel C++ Compiler.

Unsupported Compiler Options

Option	Description
`-AI<dir>`	Add to assembly search path
`-clr`	Compiler for the common language runtime
`-FU<file>`	Forced using assembly/module
`-Fx`	Merge injected code to file
`-GF`	–GF is not supported as implicit option of –O1 and –O2
`-GH`	Enable _pexit function call
`-RTCc`	Convert to smaller type checks
`-RTCu`	Un-initialized local usage checks
`-RTC1`	Enable fast checks
`-showIncludes`	Show include file names
`-w<1><n>`	Set warning level 1-4 for n
`-WL`	Enable one line diagnostics
`-wo<n>`	Issue warning n once
`-Zm<n>`	Max memory alloc (% of default)

Unsupported Major Features

• Attributed code	• Event handling (new keywords)
• Managed extensions for C++ (new pragmas, keywords, and command-line options)	
• __abstract keyword	• __box keyword
• __delegate keyword	• __gc keyword
• __identifier keyword	• __nogc keyword
• __pin keyword	• __property keyword
• __sealed keyword	• __try_cast keyword
• __typeof keyword	• __w64 keyword

Unsupported Preprocessor Features

• #import directive changes for attributed code	• #using directive
• __COUNTER__ macro	• __FUNCSIG__ macro
• _MANAGED macro	• conform pragma
• deprecated pragma	• managed, unmanaged pragmas
• runtime_checks pragma	

Differences in PCH Support
Limitations described in the section highlighting Visual Studio 6.0 compatibility are applicable to Visual Studio .Net as well.

Processor targeting
Please use the Intel C++ Compiler with Visual Studio 6.0 to develop applications for Itanium-based systems. This release of the Intel C++ compiler does not support targeting Itanium-based systems from the Visual C++ .NET environment.

References

The more information can be found from the following documents.

- *The Intel C++ Compiler for Windows Release Notes*
- *The Intel C++ Compiler for Windows User's Guide*
- *Getting Started with the Intel® C++ Compiler 7.0 for Windows*
- *Optimizing Applications with the Intel® C++ and Fortran Compilers for Windows* and Linux**